Bachelor of Software Engineering - Game Programming

GD2P02 – Physics Programming

Collision Detection - Part 1

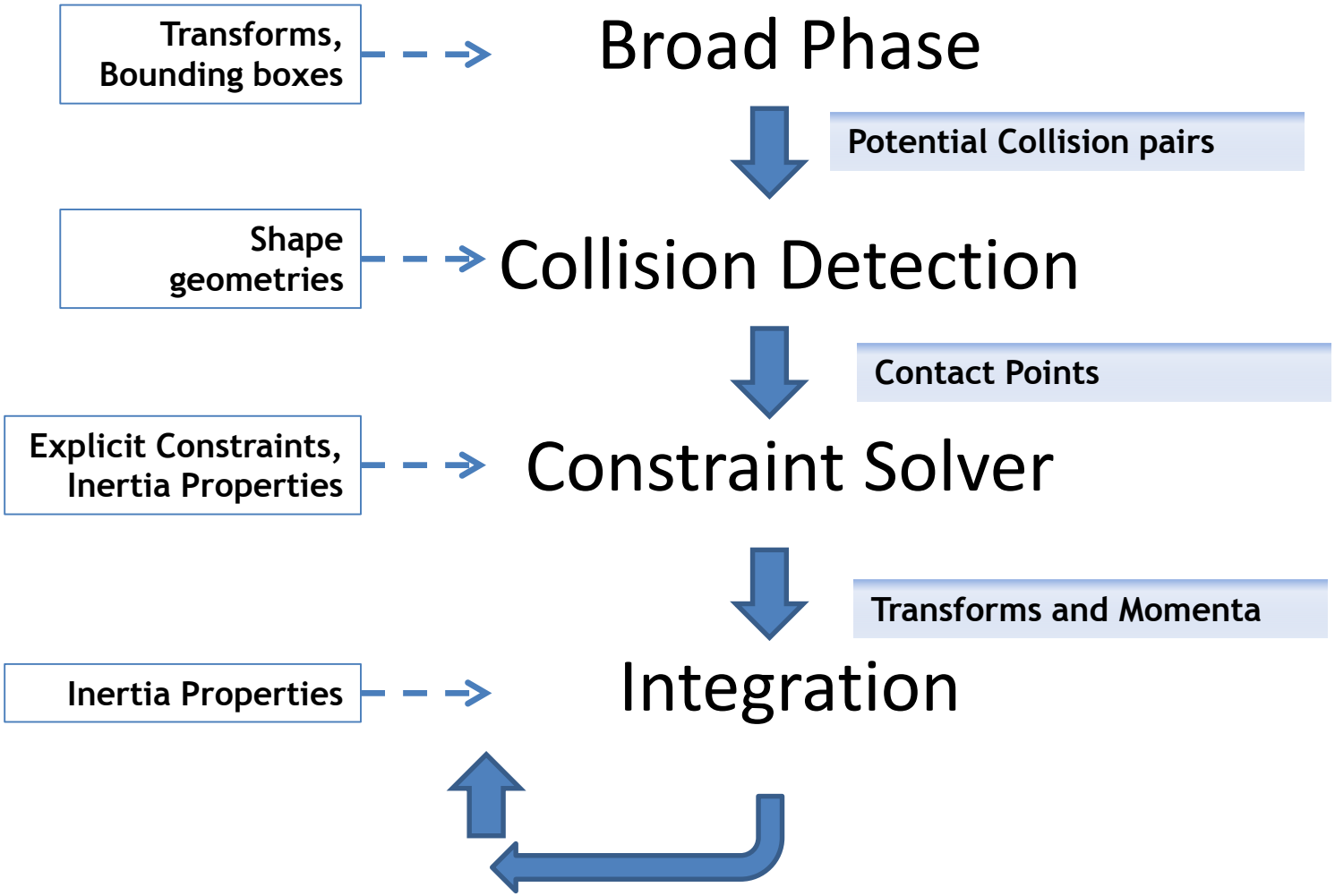
Overview

- Collision Detection
  - Physics Engine Components
  - Collision Detection
    - Collision detection pipeline
  - Discrete Collision Detection
    - Methods and Approaches
      - Bounding Spheres
      - AABB, OBB, Capsules
      - Exercise

Collision detection

- Collision detection is not just about whether there is a collision or not.
- We need the contact information.
  - Penetration (if any)
  - Contact points
  - Set of intersection points: Contact manifolds
- Computationally expensive
  - Brute Force: complexity approximately n^2
  - Methods and approaches available to reduce complexity…

Collision detection pipeline



Discrete Collision Detection (DCD) stage

- We want to know exactly how much one shape has penetrated into another and what the shortest direction to push the two apart again is.
  - Higher and lower accuracy test methods.
  - Broad Phase: Approximate and Rough.
  - Narrow Phase: Accurate, Reliable, Dependable

- General Physics Development Process:
  - Start by using Broad Phase.
  - Narrow Phase is applied to the results of Broad Phase.

Broad Phase

- Quick determination of which objects might be colliding.
  - Very approximate and rough
  - Should never fail to report a collision.

- Examples include "sweep and prune" and sphere hierarchies.

Broad Phase continued…

- No one method performs best for all situations.
- Depends on the scene conditions.
  - Indoor or outdoor scene?
  - Clustered-together or spaced-out objects?
  - Static or dynamic objects?
  - Complex or simple mesh objects?

- Trade-off between CPU cycles and extra memory usage!
  - For the complex objects, objects in motion, crowded scenes…

Broad Phase continued…

- Trade-offs:
  - CPU cycles: Computational efficiency
  - Memory footprint: Data size
  - Robustness: Reliable information
  - Complexity: Difficulty to implement or understand
  - Flexibility: Modification or extension

- Algorithms for Broad Phase:
  - Brute force
  - Sweep and Prune
  - Subdivision (QuadTrees, KD-Trees, BSP-Trees)

Bounding Spheres

- Sphere-Sphere Collision is a simple calculation both mathematically and computationally
  - We just need to know: Center position, radius
  - Check whether the distance between the spheres is less than the sum of the two radii
- Penetration depth:
  - Sum of radii minus distance between two spheres…
- Contact point:
  - Center point between the two spheres

Bounding Spheres continued…

- Avoiding the square root is a good idea.

- Quick approximation of objects
  - May need more spheres to get a more detailed approximation.

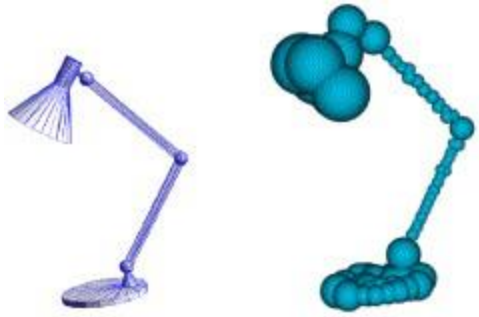


Fig 1: Left, 3D polygon lamp. Right, Sphere approximations to avoid a large bounding sphere.

Fig 1: http://www.sciencedirect.com/science/article/pii/S0097849306000665

Sphere Hierarchies

- Collisions of objects made from thousands or millions of polygons
  - And non-convex, with holes, twists…
  - Approximation with sphere hierarchies is less painful.
- Create a tree like hierarchy.
  - Whole object at tree base…
  - Leaves have pieces of the object…
  - Subdivide from the base into two pieces…
    - Subdivide these into two, and so on… until a threshold is reached.
  - Sphere boundaries calculated for each object element

Sphere Hierarchies continued…

- Trade speed for accuracy
  - Larger and less spheres runs faster: approach in Broad Phase
    - But less accurate.

- Only check child spheres if the parent is colliding!
  - Start with Broad Phase
  - Then move to Narrow Phase if required.

Sphere Hierarchies continued…

- Method:
  - Split object along axis with the longest length (x, y, or z).
  - Repeat and pass each split down to the next level, until a minimum threshold is reached.
- Triangle cutting may be handy…
- Generate hierarchies once.
  - When loading a model…
  - Update hierarchy if scaled, rotated, translated…

Other hierarchies

- Axis aligned bounding boxes (AABB)
- Oriented Bounding Boxes (OBB)
- Capsules
- Hybrid hierarchies!
  - Mixtures of shapes…
- Different approaches:
  - Inner bounds
    - Example: Circle bounds inside a square (1).
  - Outer bounds
    - Example: Circle bounds outside a square (2).

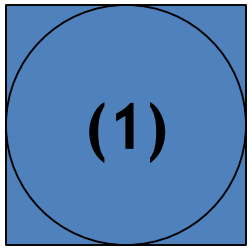


Fig 2: Inner Bounds

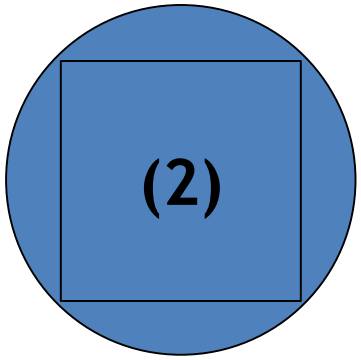


Fig 3: Outer Bounds

Capsules

- Good for representing long thin objects.
  - Character's arms
  - Poles
  - Bottles
  - Rockets
- Also known as: "Swept sphere test"
  - Same volume as a sphere travelling along a line.
- Representation:
  - Two half spheres of radius (r)
  - Connected by a tube of length (l) and radius (r)

Capsules continued…

- Capsule as line segments
  - Determine the closest point between two line segments.
  - If the distance between these two segments is less than the combined radius of the two capsules a collision has occurred.
- Two capsules:
  - Two line segments and two radii.
  - Represent line segments in parametric form!

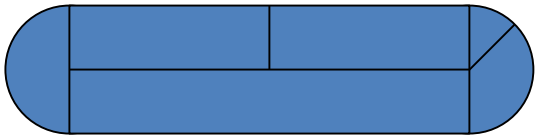


Fig 4: A capsule, showing radii.

Capsules continued…

$L_1(s) = p_1 + sN_1$

$L_2(t) = p_2 + tN_2$

– Subtract them from one-another:

$v(s,t) = L_2(t) – L_1(s)$

- v(s,t) is a direction vector…
- L_1 and L_2 are position vectors…
- N_1 and N_2 represent directions of the lines…

– v(s,t) is the shortest path between the two lines!

Capsules continued…

- If we take any random point in space…
  - The closest point between the point and the line will always be orthogonal!

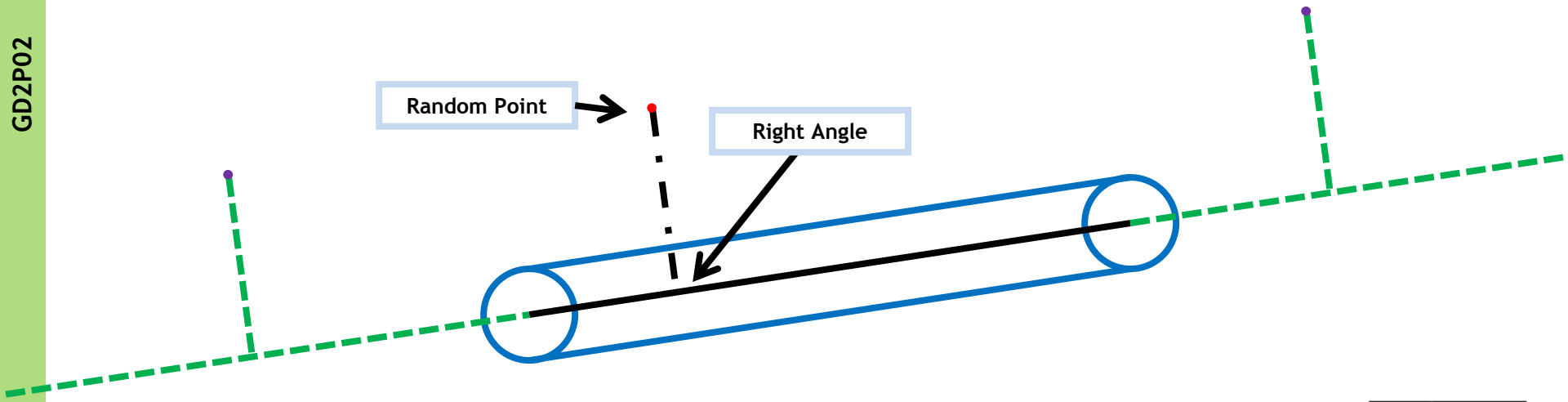


Fig 5: A capsule vs point.

Capsules continued…

- Extend this idea further…
  - Two points, two lines…
  - With:
    - The closest point to each line is located on the opposite line.
    - These two lines will be parallel.
    - Shortest path must be orthogonal to both lines.
- Remember:
  - The dot product between two orthogonal vectors is zero.
- Therefore:

$$N_1 \bullet v(s,t) = 0$$

$$N_2 \bullet v(s,t) = 0$$

Capsules continued…

- Substitute and solve for s and t:

$$s = ((N_1 \bullet N_2)(N_2 \bullet r) - (N_1 \bullet r)(N_2 \bullet N_2)) / d$$

$$t = ((N_1 \bullet N_1)(N_2 \bullet r) - (N_1 \bullet N_2)(N_1 \bullet r)) / d$$

- Limit s and t between 0 and 1…

- Now we have two line segments, and two points…
- We must find the point on the line segment that is closest to the point…
  - Then we can do a radius check…

Capsules continued…

- Use the dot product and projection
  - Project the point onto the line to get the closest point.

- There are three different dot product cases for projection of the point onto the line segment…

$$t = (P - A) \bullet n$$

Where: $n = (B - A)/(|B - A|)$

Therefore: $t = ((P - A) \bullet (B - A)) / (|B - A|)$

Capsules continued…

Three different cases for projecting the point onto the line segment:

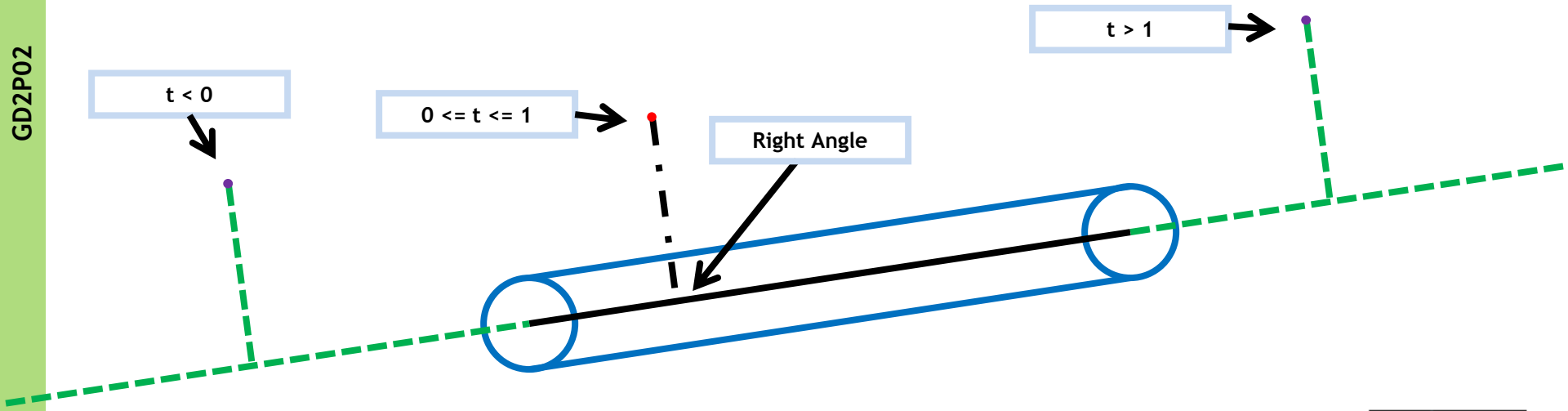


Fig 6: Projecting the point onto the line, with resulting t value.

Capsules continued…

- The point on the line segment D, which is closest to the point P is:

$$D = A + tn$$

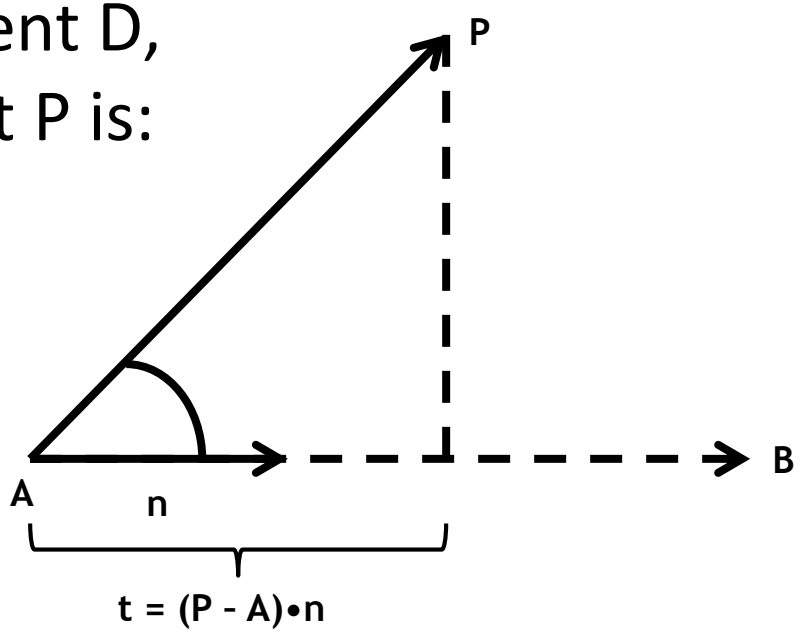


Fig 7: Closest point to a line segment.

- Capsule vs Capsule…
- If the distance between these two segments is less than the combined radius of the two capsules, a collision has occurred.

Axis Aligned Bounding Boxes (AABB)

- Rectangular cubes aligned to the world axis
  - Cuts down the transformation cost to world coordinates
  - However, we cannot just rotate them when the object rotates.
    - They need to be recomputed for each frame.
  - There are precision issues.
  - May not be tightly enclosing.

Oriented Bounding Boxes (OBB)

- Rectangular cubes that are arbitrarily oriented
  - Similar to AABB, but not axis aligned!
  - Tight fitting
  - OBBs move and rotate together with the bounded object.
- OBB vs OBB: Computationally more expensive
  - Difficult to implement, slower, not suitable for dynamic or procedural models
- Separating Axis Theorem can be applied with OBBs too.
  - SAT will work with convex shapes, not just cubes.

Narrow Phase

- Gives accurate collision detection information.
  - Real shapes are tested against each other.
  - Expensive test.
  - Vital information for collision response phase is obtained.
    - Collision point
    - Normal
    - Distance/Penetration depth
- Algorithms for Narrow Phase:
  - Triangle-triangle
  - Separating Axis Theorem (SAT)
  - Gilbert-Johnson-Keerthi distance algorithm (GJK)

Exercise 003.1 - 2D Capsule vs 2D Capsule

- Closest point to line segment
- User can define two capsules:
  - Defining a new capsule:
    - First Click: Center of one end.
    - Second Click: Radius of the end.
    - Third Click: Center of the second end.

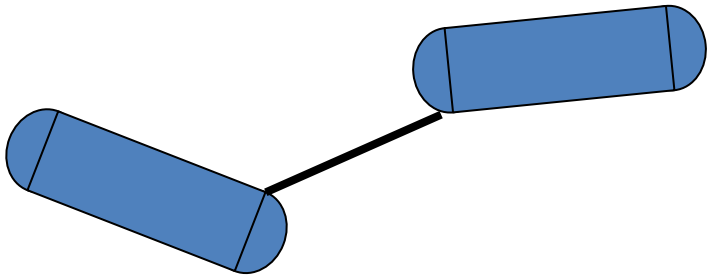


Fig 18: Example capsules.

- Determine the shortest distance between the two capsules.
- Allow the scene to be reset:
  - The R key.

Summary

- Collision Detection
  - Physics Engine Components
  - Collision Detection
    - Collision detection pipeline
  - Discrete Collision Detection
    - Methods and Approaches
      - Bounding Spheres
      - AABB, OBB, Capsules
      - Exercise